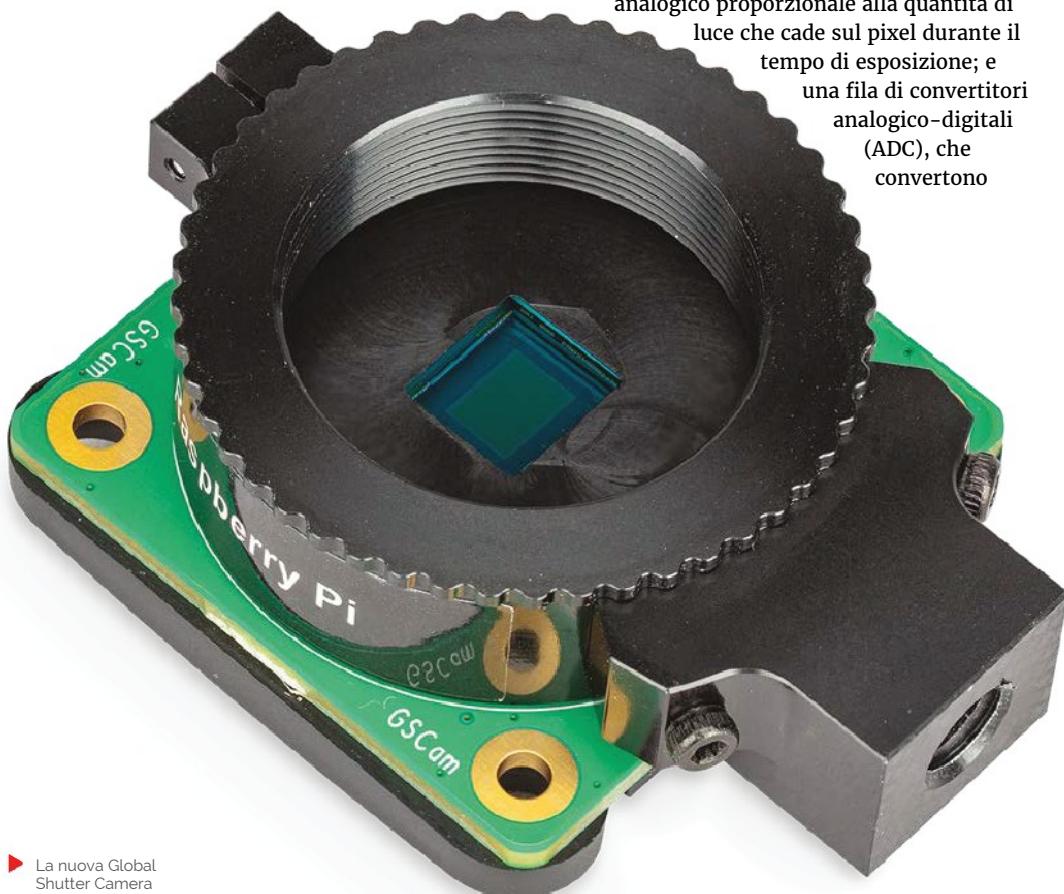
Numero 128 | Aprile 2023 | magpi.cc raspberryitaly.com

La rivista ufficiale Raspberry Pi tradotta in italiano per RaspberryItaly

IMPARA A PROGRAMMARE CON PYTHON

Scopri la prpgrammazione con Raspberry PI

NUOVA! RASPBERRY PI GLOBAL SHUTTER CAMERA

Global Shutter Camera

Nuova fotocamera Raspberry Pi Global Shutter per visione artificiale e altro ancora. Di **Eben Upton**

Ricordi quelle nuove fotocamere che abbiamo lanciato all'alba di... gennaio? [Vedi bit.ly/MagPi126It – Ndr]. Ne abbiamo un'altra per te – e questa volta ci stiamo specializzando. Stiamo lanciando qualcosa di leggermente diverso: la Raspberry Pi Global Shutter Camera, disponibile ora a 50$ (**magpi.cc/globalshuttercam**).
Costruita attorno al sensore IMX296 da 1,6 megapixel di Sony, la Global Shutter Camera è in grado di catturare movimenti rapidi senza introdurre artefatti da otturatore. Questo la rende ottima per la fotografia sportiva e le applicazioni di visione artificiale, dove anche piccole distorsioni possono degradare seriamente in inferenza.

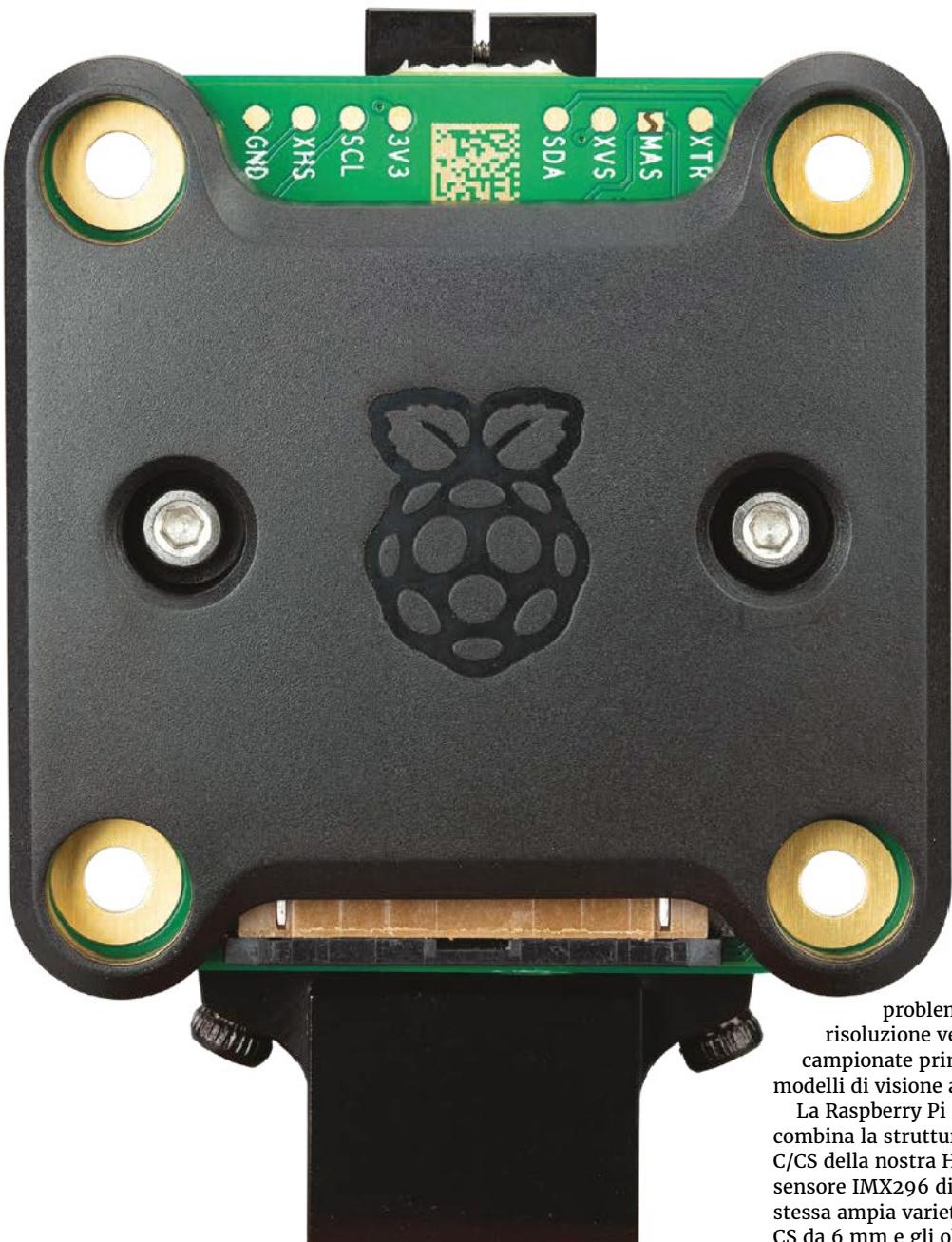
Otturatore rotante, otturatore globale

Ogni fotocamera che abbiamo rilasciato fino ad oggi, da Camera Module 1 del 2014 alla nostra High Quality Camera e non solo, ha utilizzato un sensore con otturatore scorrevole. Questi sensori hanno una matrice bidimensionale di pixel sensibili alla luce, che generano un valore analogico proporzionale alla quantità di luce che cade sul pixel durante il tempo di esposizione; e una fila di convertitori analogico-digitali (ADC), che convertono

La nuova Global Shutter Camera

questi valori analogici in valori digitali che vengono restituiti al Raspberry Pi.

La fila di ADC è collegata a sua volta a ciascuna riga dell'array di pixel, quindi ogni riga viene campionata in un momento leggermente diverso. A condizione che non ci sia movimento nella scena, questo non è un problema, ma una volta che le cose iniziano a muoversi – e soprattutto se si muovono velocemente - iniziamo a vedere artefatti dovuti allo otturatore. Il moto lineare provoca compressione, stiramento, o taglio dell'oggetto in movimento. Il movimento rotatorio può creare molte forme davvero strane. Gli artefatti degli otturatori rotanti sono antiestetici e difficili da rilevare e correggere, ma artefatti anche impercettibili possono

> Quando scatta l'otturatore, ogni pixel immediatamente copia il suo valore analogico nel suo spazio di storage

interferire con il funzionamento degli algoritmi di visione artificiale. Se vogliamo eliminarli del tutto, dobbiamo usare un sensore con un otturatore globale. Questo accoppia ogni pixel con un elemento di archiviazione analogico; quando l'otturatore si attiva, ogni pixel copia immediatamente il suo valore analogico nel suo elemento di archiviazione, da dove può essere letto e convertito a piacere.

L'elemento di archiviazione aggiunge complessità e area ad ogni pixel. I sensori con otturatore globale tendono ad avere una risoluzione inferiore rispetto ai sensori normali con le stesse dimensioni: contrastano il 7,9 mm, 12 megapixel del sensore IMX477 utilizzato nella High Quality Camera con l'IMX296 da 6,3 mm e 1,6 megapixel.

▲ La Global Shutter Camera combina l'attacco C/CS della High Quality Camera con il sensore Sony IMX296

Crediti

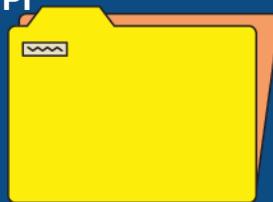
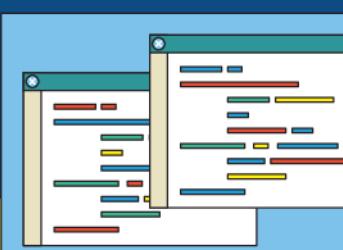
Come tutti i recenti prodotti per fotocamere, l'hardware della Global Shutter Camera è stato progettato da Simon Martin. Naush Patuck, Nick Hollinghurst, David Plowman, Serge Schneider e Dave Stevenson hanno scritto il software. Alasdair Allan, Simon Martin, David Plowman, Andrew Scheller e Liz Upton hanno lavorato sulla documentazione. Austin Su ha guidato l'approvvigionamento. Jack Willis ha disegnato la confezione e Brian O Halloran ha scattato le foto e video (non inclusi con la fotocamera). Ringraziamo per l'assistenza Phil Holden e John Conroy della Sony, e della Shenzhen O-HN Optoelectronic.

Se pensi che la mia parte anteriore sia bella...
c'è un logo a sorpresa per i fan della vista posteriore

Introdurre la Global Shutter Camera di Raspberry Pi

Nonostante le sfide associate agli artefatti dell'otturatore rotante, le nostre telecamere sono ampiamente usate in applicazioni hobbistiche e di visione artificiale. E abbiamo visto una cosa davvero ingegnosa: per compensare gli artefatti durante l'imaging dei prodotti su un nastro trasportatore ad alta velocità, uno dei nostri clienti industriali ha finito per addestrare i propri modelli utilizzando dati di input pre-tagliati.

Per queste applicazioni, un sensore con otturatore globale offre evidenti vantaggi. E la risoluzione ridotta non è un problema, poiché le immagini ad alta risoluzione vengono generalmente sotto campionate prima di essere inserite nei modelli di visione artificiale.

La Raspberry Pi Global Shutter Camera combina la struttura in metallo con l'attacco C/CS della nostra High Quality Camera con il sensore IMX296 di Sony. È compatibile con la stessa ampia varietà di obiettivi, inclusi l'attacco CS da 6 mm e gli obiettivi CGL con attacco C da 16 mm che offriamo attraverso i nostri rivenditori partner approvati.

Come tutti i nostri prodotti per fotocamere, puoi utilizzare la Global Shutter Camera con qualsiasi computer Raspberry Pi dotato di connettore per fotocamera CSI e abbiamo aggiornato la documentazione hardware (**magpi.cc/cameradocs**) per includere tutto il necessario per conoscere il nuovo prodotto. Devi aggiornare Raspberry Pi OS per utilizzare la nuova fotocamera: `sudo apt update; sudo apt full-upgrade; sudo reboot` e sei a posto.

Camera in azione

Il video su **magpi.cc/gscamaction** illustra i vantaggi di un otturatore globale in presenza di movimento di rotazione rapida. Per prima cosa, vediamo l'output della High Quality Camera, che mostra artefatti distintivi dell'otturatore rotante e poi vediamo l'output privo di artefatti del Global Shutter Camera.

IMPARARE A PROGRAMMARE CON PYTHON

Scopri la gioia della programmazione con la nostra guida per principianti a Python 3 su Raspberry Pi

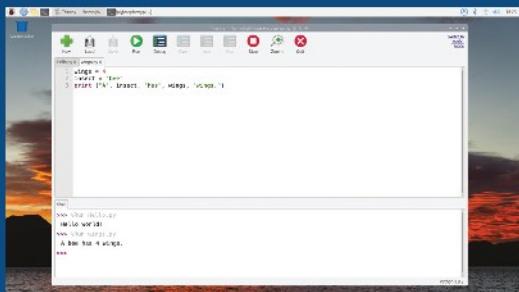
di Phil King

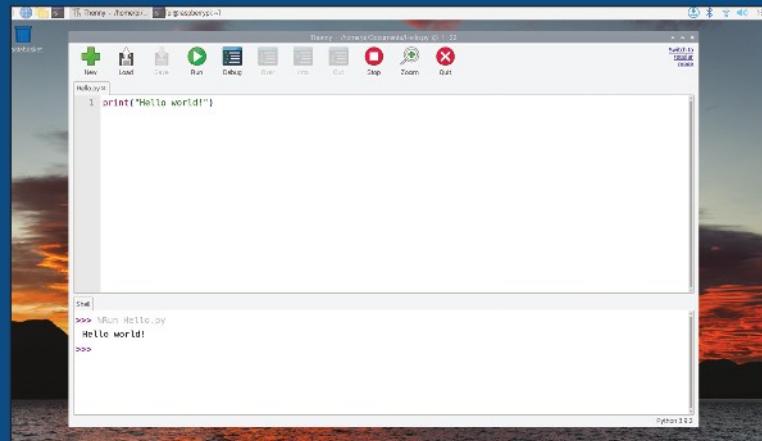
In questo corso accelerato, ti guideremo ai principi base della programmazione Python su Raspberry Pi.

Per questa guida, scriveremo il codice nell'IDE (ambiente di sviluppo integrato) Thonny del sistema operativo Raspberry Pi. Un IDE è un programma utilizzato per creare altri programmi. Puoi creare programmi in qualsiasi editor di testo, ma l'utilizzo di un IDE semplifica la vita (soprattutto per i nuovi arrivati).

Dal menu delle applicazioni, seleziona Programmazione > Thonny. L'IDE Thonny si avvia in modalità semplice per default. Lungo la parte superiore ci sono icone grandi tra cui Carica, Salva, Esegui e Interrompi.

Scriviamo un po' di codice

Il pannello principale è dove scriverai le linee del codice Python. In fondo c'è il pannello Shell dove vedrai l'output e gli eventuali messaggi di errore risultanti da un programma in esecuzione.

Scriviamo un po' di codice!

Per il nostro primo programma, stamperemo un messaggio nella Shell. Questa è la finestra di Thonny in basso che visualizza i risultati (o gli output) del nostro programma. È tradizione dare il benvenuto ad un nuovo linguaggio di programmazione facendogli dire "Ciao mondo!"

Nel pannello principale di Thonny, sulla riga 1, aggiungi il seguente codice Python:

```
print("Ciao, Mondo!")
```

Le doppie virgolette sono utilizzate per racchiudere un messaggio di testo (noto come stringa). Fai clic sull'icona Esegui e Thonny eseguirà e stamperà il messaggio nella Shell. Prova a cambiare il testo tra le doppie virgolette e riesegui per vedere un messaggio diverso.

▲ I numeri e le stringhe di testo possono essere memorizzati in variabili e quindi recuperati quando necessario

▲ Ciao mondo: il più semplice dei programmi è stampare un messaggio nell'area della Shell

È possibile salvare il programma facendo clic su Salva e inserendo **hello_world.py** nel campo Nome.

Tipi di dati e operatori matematici

Nel nostro programma **hello_world.py**, abbiamo stampato una stringa di testo. Altri tipi di dati in Python includono numeri interi (come 1, 2 e 3) e numeri in virgola mobile (decimali come 3.5 o 10.532).

Per crearli, omettiamo le virgolette doppie nella nostra dichiarazione di stampa. Fai clic su Nuovo per iniziare un nuovo programma e digita:

PERCHÉ PYTHON?

Ci sono molti linguaggia di programmazione diversi, ma Python è di gran lunga il più popolare per i principianti. Ecco cinque motivi per cui probabilmente dovresti scegliere Python come primo linguaggio:
1. **È semplice**: Python ha una sintassi relativamente semplice e di facile apprendimento.
2. **Python è versatile**: Python può essere utilizzato per una vasta gamma di applicazioni, dallo sviluppo web all'analisi dei dati e intelligenza artificiale.
3. **Comunità ampia e attiva**: Python ha una comunità di sviluppatori numerosa e attiva che contribuiscono al suo sviluppo. Questa comunità garantisce che ci siano molte risorse disponibili per l'apprendimento e la risoluzione dei problemi.
4. **Open-source:** Python è un linguaggio open-source, il che significa che il codice sorgente è liberamente disponibile per chiunque lo voglia utilizzare, modificare, e distribuire.
5. **Python è portabile**: codice scritto in Python può essere eseguito su più piattaforme tra cui SO Raspberry Pi, altre distribuzioni Linux, Windows e macOS.
Tutte gli altri linguaggi offrono vantaggi ma noi pensiamo che Python sia un ottimo punto di partenza.

Tip!

Nominare il tuo programma

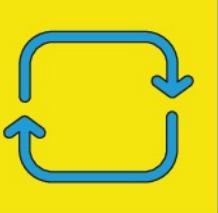
I programmi Python sono tipicamente nominati con tutti caratteri minuscoli con sottolineature multiple per scindere le parole, cioè:
- gioco.py
- ciao_mondo.py
- generatore_numero.py

È buona abitudine assicurarti che il nome del file rifletta quel che fa il programma. Prova a evitare spazi e nomi generici.

Le variabili possono essere settate per ricevere inserimenti dall'utente; per pulizia, includere uno spazio dopo il testo

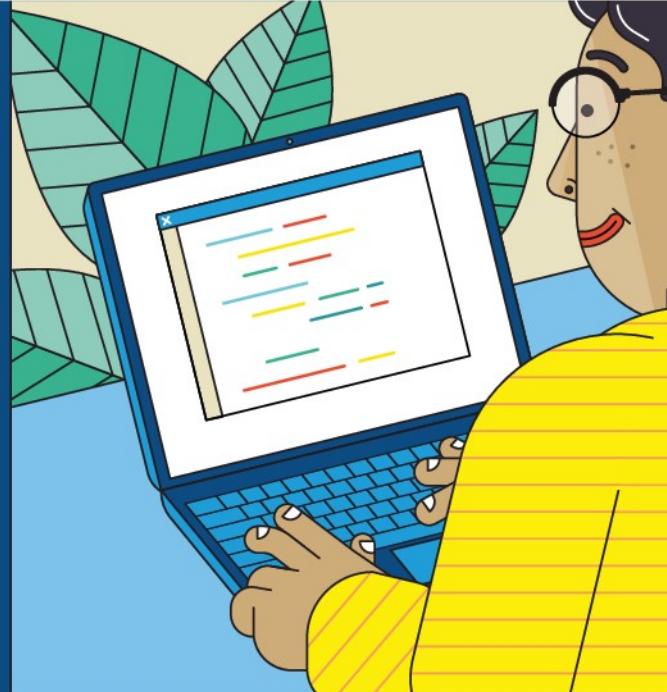
```
print(128)
```

Fai clic su Esegui per stampare il numero intero 128 nella Shell. Possiamo anche usare operatori matematici standard (addizione, sottrazione, moltiplicazione e divisione) sui numeri. Per esempio:

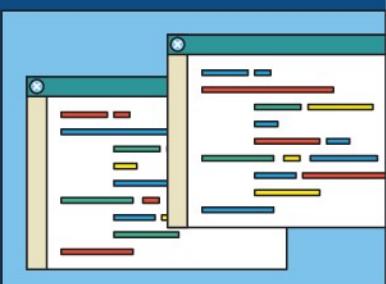
> **Memorizziamo numeri e stringhe in oggetti chiamati variabili**

```
print(128+57)
```

Questo produrrà il risultato 185. Per la moltiplicazione e la divisione si usano i simboli * e /. Per esempio:

```
print(128/8+5*7)
```

...produce il risultato 51.0. Ci sono due cose da notare qui. Innanzitutto, l'operatore di divisione fornisce automaticamente un risultato in virgola mobile (quindi la cifra decimale). In secondo luogo, si applica la regola BODMAS/PEMDAS, quindi la divisione e la moltiplicazione sono eseguite prima dell'addizione e della sottrazione.

L'operatore di addizione può essere utilizzato anche con le stringhe. Per esempio:

Test multipli condizionali con dichiarazioni if e elif; la seguente riga rientrata viene eseguita solo se la condizione è vera

```
print("The MagPi" + " è " + "una rivista.")
```

Nota gli spazi attorno alla parola "è". Ancora, gli operatori matematici includono ** per "alla potenza", % per modulo (resto della divisione), e // per divisione floor (arrotondata per difetto all'intero più vicino). Per esempio:

```
print(2**4)
print(20%7)
print(22//7)
```

Che danno risultati 16, 6 e 3 rispettivamente.

Variabli

Memorizziamo numeri o stringhe in oggetti così chiamati variabili – così chiamati perché possiamo cambiare (o 'variare') il loro valore nel nostro programma.

CAMBIARE TIPO DI DATI

È possibile scambiare numeri o stringhe tra tipi di dati se necessario. Per passare da un numero a una stringa, usiamo la funzione `str()`. Per esempio:

```
x = 42
message = "Il senso della vita è "
+ str(x)
print(message)
```

Per passare da un tipo di numero all'altro (o convertire una stringa composta da un numero), usiamo le funzioni `int()` e `float()`.

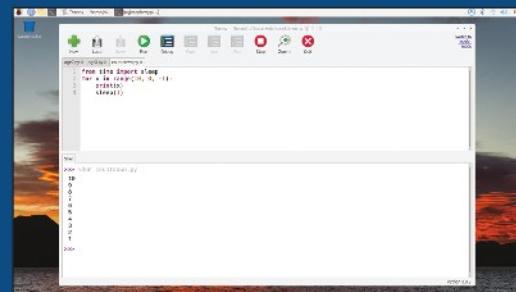
OPERATORI DI COMPARAZIONE

In Python, sono disponibili sei operatori di comparazione:

Operatore	Significato
==	uguale a
!=	diverso da
>	maggiore di
<	minore di
>=	maggiore o uguale a
<=	minore o uguale a

Nota che devi utilizzare == per le comparazioni, non = (che è usato per settare un valore).

Per definire una variabile, scegliamo un nome per essa e usiamo il simbolo = per settare il suo valore. Per esempio:

```
wings = 4
animal = "ape"
print("Un", animal, "ha", wings, "ali.")
```

Usando la funzione predefinita on Python `input()` otteniamo l'input dell'utente e lo memorizziamo in una variabile:

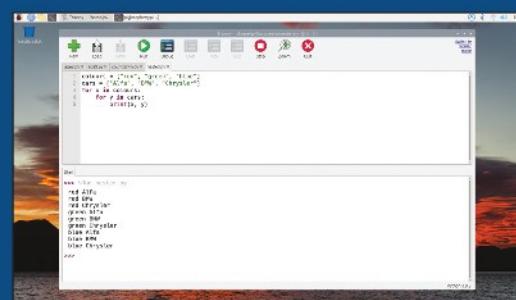
```
name = input("Come ti chiami? ")
age = input("Quanti anni hai? ")
print("Ciao " + name + ", hai " + age +
" anni di età.")
```

Condizioni

Una parte fondamentale di molti programmi comporta il confronto tra variabili e valori per determinare quale passo dovrà essere eseguito successivamente. Per questo, usiamo una dichiarazione condizionale `if`.

```
age = input("Quanti anni hai? ")
age = int(age)
if age > 12 and age < 20:
    print("Sei un teenager.")
```

Usare un ciclo for Per fare il conto alla rovescia da dieci a uno, con un ritardo sleep tra i numeri

La riga dopo la condizione `if` è identata di quattro spazi e il codice indentato viene eseguito solo se la condizione è vera (es la variabile `age` è tra 13 e 19). In questo esempio viene utilizzato un operatore logico `and` per verificare che due condizioni siano entrambe vere.

Possiamo aggiungere una condizione `elif` (abbreviazione di 'else if') successivamente per eseguirne un altro confronto se nessuno dei precedenti è vero:

OPERATORI LOGICI

Sono usati per comparare multiple condizioni in una volta.

Operatore	Significato
and	entrambe le condizioni vere
or	una (o entrambe) le condizioni sono vere
nor	Entrambe le condizioni flase

Nota che l'operatore `or` in Python è non-esclusivo, quindi è ancora vero se entrambe le condizioni sono vere. Altri tipi di operatore disponibili in Python includono identità (`is`, `is not`), appartenenza (`in`, `not in`) e bit per bit (per confrontare i numeri binari).

Utilizzo di nidificato dei loop per stampare elementi da due liste diverse

Un dizionario include coppie di chiavi e valori, separate da due punti

```
elif age <= 12:
    print("Sei un bambino.")
```

Puoi aggiungerne quante condizioni `elif` vuoi. Alla fine, puoi anche aggiungere una condizione `else` che eseguirà la riga successiva solo se nessuna delle condizioni precedenti è vera.

> **Una lista serve per memorizzare una sequenza ordinata di valori**

```
else:
    print("Sei un adulto.")
```

Cicli

Invece di scrivere righe e righe di codice per eseguire più iterazioni, puoi usare un ciclo `for` Per ripeterle con valori diversi all'interno una gamma, un range.

```
for x in range(10):
    print(x)
```

Questo stampa le cifre da 0 a 9, dal valore di partenza predefinito per la funzione `range()`,

Stampare i valori delle tuple memorizzati in un elenco

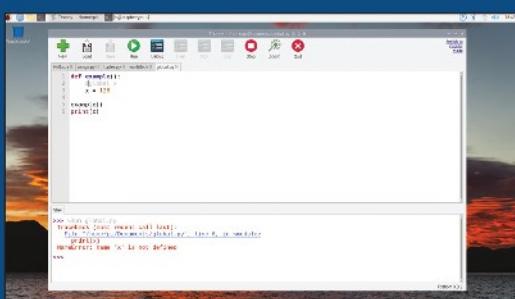
cioè zero. Il valore 10 che impostiamo è quello di stop. Per farlo contare da 1 a 10, dobbiamo usare

```
for x in range(1, 11):
    print(x)
```

Per il conto alla rovescia, invece, regola i parametri di inizio e fine del `range()` e aggiungine un terzo per cambiare la dimensione del passo in -1.

```
for x in range(10, 0, -1):
    print(x)
```

Se vuoi aggiungere un ritardo tra la stampa ogni numero, puoi usare la funzione `sleep` dal modulo time di Python, ma prima la devi importare.

CICLO WHILE

Un'alternativa al cilco `for` è usare `while` per testare una condizione e eseguire il loop solo se è vera.

```
x = 1
print("Conto da 1 a 5…")
while x <= 5:
    print(x)
    x = x +1
```

Possiamo usare anche `while True` per creare un ciclo che viene eseguito all'infinito (finché non fermi il programma), poiché la condizione è sempre vera.

L'area Shell mostra eventuali errori durante l'esecuzione del codice; qui non è riuscito a fornire l'ambito globale variabile di una funzione

```
from time import sleep
for x in range(10, 0, -1):
    print(x)
     sleep(1)
```

Un'altra cosa che puoi fare con i cicli è nidificarli, cosa che tratteremo nella prossima sezione.

Liste

In Python, una lista viene utilizzata per memorizzare una sequenza ordinata di valori (di qualsiasi tipo) separati da virgole. È quindi possibile accedervi individualmente utilizzando un numero di indice.

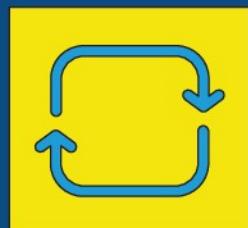
```
names = ["Alex", "Brooke", "Chloe",
"David", "Ed"]
print(names[1])
```

Poiché Python inizia a contare da 0, l'indice 1 (riportato tra parentesi quadre) seleziona "Brooke" dalla lista.

> "Verrà usata una lista con un ciclo for"

In genere, una lista verrà utilizzata con un ciclo `for`, per stampare una gamma di valori.

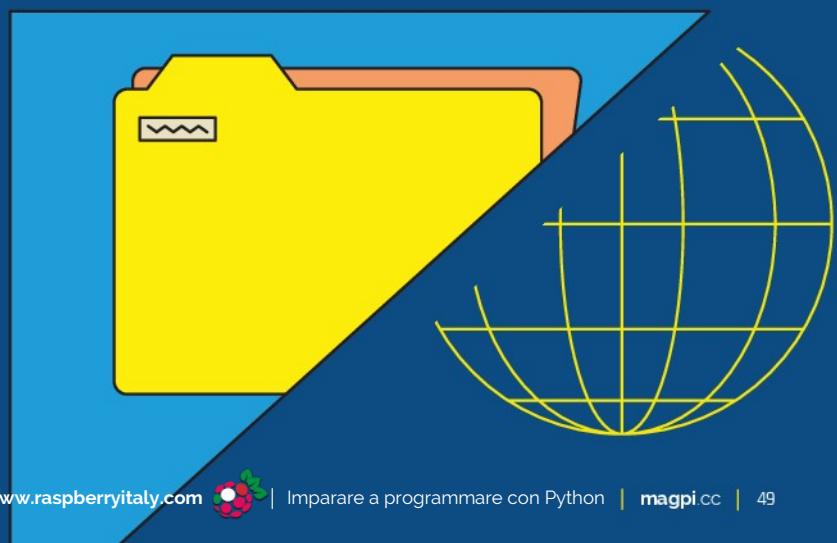
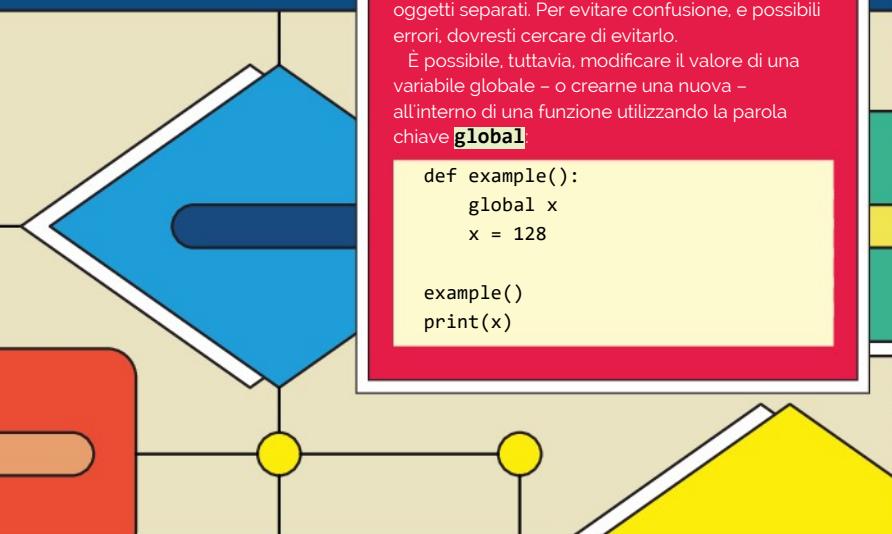
```
fruits = ["mela", "banana", "ciliegia",
"susina", "sambuco"]
for i in range(5):
    print(fruits[i])
```

Possiamo anche nidificare i cicli per scegliere gli elementi due o più liste.

```
colours = ["rosso", "verde", "blu"]
cars = ["Alfa", "BMW", "Chrysler"]
for x in cars:
    for y in colours:
        print(x, y)
```

VALORI DI RITORNO

Invece di usare `print()` nella nostra funzione di addizione, potremmo usare `return` per ritornare il risultato al chiamante

```
def add(a, b):
    return a + b

result = add(138, 485)
print(result)
```

Vengono stampate nove righe, da "Alfa rossa", "BMW rossa", "Chrysler rossa", "Alfa verde"...

Nota che qui, a differenza della funzione `range()` per impostare un indice, stiamo usando l'operatore di appartenenza `in` per selezionare ogni elemento a turno della lista. Una lista può anche contenere tuple. Una tupla è una serie indicizzata di valori separati da virgola (che non possono essere modificati) racchiusi tra parentesi.

```
rgb = [(255, 0, 0), (0, 255, 0), (0, 0,
255)]
colours = ("rosso", "verde", "blu")
for i in range(3):
    print("Il valore RGB per il", colours[i],
"è", rgb[i])
```

Dizionari

Sono simili alle liste tranne che contengono una serie di coppie "chiave:valore" separate da due punti, racchiuse tra parentesi graffe. Invece di utilizzare un indice, usiamo la chiave per accedere al valore corrispondente.

AMBITO DELLE VARIABILI

Una variabile creata all'interno di una funzione è disponibile solo all'interno di quella funzione, o di qualsiasi funzione annidata al suo interno. Questo è noto come variabile locale.

Al contrario, a una variabile creata all'interno del programma Python principale si pu avere accesso da ovunque, anche nelle funzioni. Questo è noto come variabile globale.

Puoi persino creare variabili globali e locali con lo stesso nome, che saranno trattate come oggetti separati. Per evitare confusione, e possibili errori, dovresti cercare di evitarlo.

È possibile, tuttavia, modificare il valore di una variabile globale – o crearne una nuova – all'interno di una funzione utilizzando la parola chiave `global`:

```
def example():
    global x
    x = 128

example()
print(x)
```

```
phonebook = {
    "Alex" : "0123456789",
    "Bryony" : "9876543210",
    "Charlie" : "0246813579"
    }
name = input("Di chi vuoi il numero di telefono? ")
print(phonebook[name])
```

Nota che poiché gli interi decimali non possono avere uno zero iniziale in Python, stiamo usando delle stringhe per i numeri di telefono.

Funzioni

Una funzione Python è solo un blocco di codice che viene eseguito quando "chiamato". Python ha molte funzioni integrate, ma puoi anche definirne di tue personalizzate.

Usare le funzioni nei tuoi programmi è un buon modo per organizzare il tutto, semplifica la lettura del tuo codice da parte di altri, ed evitare di dover scrivere ripetitivi blocchi di codice simile.

Per definire una funzione, usiamo `def` seguito da un nome, parentesi (che possono contenere uno o più argomenti) e due punti. Questo è seguito dalle righe di codice rientrate della funzione. Ad esempio, una semplice funzione per sommare tra loro due numeri:

```
def add(a, b):
    result = a + b
    print("La somma è", result)
```

Ora che abbiamo definito la funzione, possiamo richiamarla con il suo nome seguito da parentesi che contengono i valori per gli argomenti (se presenti). Per esempio:

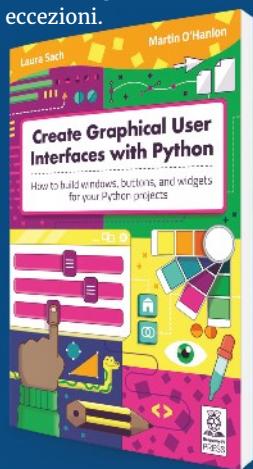
```
add(138, 485)
```

Puoi mettere questa riga in un ciclo `while True:` e chiedere all'utente di inserire i numeri da sommare.

Quando guardi i programmi Python, vedrai spesso funzioni definite e chiamate senza eventuali argomenti.

```
def message():
    print("Questo è il mio messaggio!")

message()
```

Questo chiamerà semplicemente la funzione per eseguirla e, in questo caso, scrivere un messaggio.

Molto altro da imparare

Speriamo che questa guida aiuti i nuovi arrivati a iniziare a programmare in Python, ma in realtà qui ne abbiamo solo scalfito la superficie. Ci sono molti altri aspetti del linguaggio, come l'uso della formattazione delle stringhe, classi e oggetti, numpy array, espressioni regolari, decoratori e gestione delle eccezioni. Puoi anche aggiungere della grafica ai tuoi Programmi, usando moduli come Pygame o Pygame Zero e una GUI Con strumenti come Tkinter o guizero.

Prendi una copia di *Create Graphical User Interface with Python* per continuare il tuo viaggio in Python. Buona programmazione!

ASTROFOTOGRAFIA con Raspberry Pi

Ottieni incredibili scatti del cosmo con il tuo Raspberry Pi, un Camera Module e un po' di codice smart. By guida stellare Rob Zwetsloot

Guardare il cielo notturno è qualcosa che facciamo fin dai tempi antichi, con stelle, pianeti e lune che brillano dal cosmo per ricordarci che c'è un intero universo oltre la Terra. Scattare foto del cielo notturno per catturare queste viste, può però essere un po' complicato.

È qui che entra in gioco Raspberry Pi. Con una gamma di eccellenti moduli fotocamera e automazione personalizzabile, puoi facilmente creare una configurazione che catturerà le meraviglie della notte e la bellezza dei lontani corpi celesti. È tempo di ascoltare il richiamo della notte.

Le basi fotografare il cielo

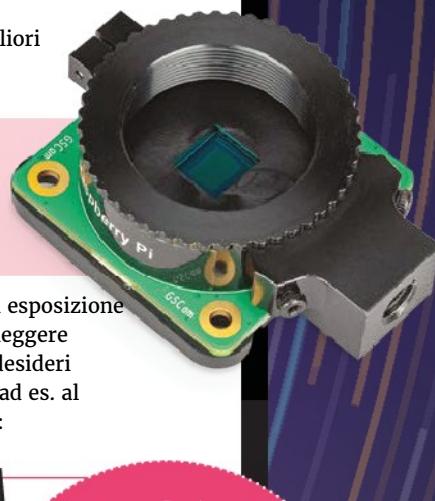
Ottieni i migliori scatti cosmici con questi suggerimenti fuori dal mondo

Puoi ottenere ottimi scatti anche senza un telescopio, usando solo una Raspberry Pi HQ Camera, se scegli gli obiettivi giusti e programmi lo scatto in un modo specifico.

01 Scegliere un obiettivo

Senza un telescopio, probabilmente miri a foto del cielo notturno che includano molto del cielo e non a concentrarti sui corpi celesti. In questo caso, gli obiettivi grandangolari 16-24 mm Sono una buona opzione, il che significa che l'obiettivo da 16mm spesso venduto con Le HQ Camera è un perfetto punto di partenza.

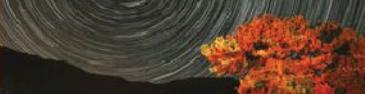
02 Stabilire l'esposizione

Un'esposizione più lunga consente al sensore di catturare più luce. Per l'astrofotografia, normalmente occorrono **da 15 a 30 secondi** di esposizione per catturare sufficienti dati luminosi. La libreria libcamera è programmata in microsecondi, quindi questi valori sarebbero 15.000.000 e 30.000.000, rispettivamente.

03 Codice a linea di comando

Con l'obiettivo e le giuste conoscenze delle impostazioni di esposizione, possiamo iniziare a costruire un comando che usi libcamera. Per lunghe esposizioni, dobbiamo disabilitare alcuni algoritmi di controllo automatico per i migliori risultati. Nel complesso, il comando si presenta così:

```
libcamera-still -o
nightsky.jpg --shutter
15000000 --gain 1
--awbgains 1,1 -
immediate
```

Questo imposta 15 secondi di esposizione come JPG di alta qualità. Puoi leggere ulteriormente i documenti se desideri apportare ulteriori modifiche, ad es. al guadagno del livello di cattura: **magpi.cc/libcamera**.

The HQ Camera ha un tempo di esposizione massimo di circa dieci minuti

Tip

La HQ Camera utilizza attacco C/CS e M12 per gli obiettivi, in modo da poter utilizzare qualunque obiettivo ti piaccia, se è compatibile

68 Creare foto "star trail"

70 Scattare con un telescopio

Creare foto "star trail"

Use long exposure and time-lapse techniques to create beautiful night shots

Con la configurazione di base, puoi creare un paio di tipi di classici scatti spaziali - foto meravigliose del cielo notturno o eleganti scie stellari. Con uno Script Python, puoi creare la scia delle stelle, effetto creato dalla rotazione terrestre.

Hardware

Drive USB

Scattare un sacco di grandi immagini può occupare spazio davvero velocemente

Schermo (opzionale)

È sempre una buona idea verificare che foto stai scattando, anche se puoi sempre usare VNC se non hai uno schermo

Raspberry Pi (qualsiasi)

Puoi usare qualsiasi Raspberry Pi con una porta CSI, che Significa la maggior parte dei Raspberry Pi Zero e qualsiasi Modello standard. Per più è moderno, e meglio è.

Raspberry Pi High Quality Camera e obiettivi

Vedi la pagina precedente per consigli su quale obiettivo prendere - sebbene il 16 mm potrebbe andare più che bene per questo

Power bank

Lo vorrai fare all'esterno.. Quindi una batteria portatile decente è essenziale

Software

Un po' di semplice codice Python che puoi usare per ottenere una notte piena di scatti. Mettilo sulla tua chiavetta USB

A seconda di quanto è grande l'unità USB che hai collegato, puoi sempre aumentare il numero di foto e il tempo di esposizione (fino a 60 secondi), e diminuire il numero di secondi tra uno scatto e l'altro!

01 Impostare il software

Per fare una sequenza di foto, useremo GNU Image Manipulation Program (GIMP) da **gimp.org**. Dopo averlo installato, avrai anche bisogno di un plug-in per creare lo stack di foto. Prendilo da **magpi.cc/starstack** e segui le istruzioni.

02 Ottieni le tue foto

Prendi l'unità USB con le tue foto e inseriscila nel tuo PC. Suggeriamo di copiarla in una nuova cartella sul tuo PC con cui lavorare – assicurati di trovarla facilmente! Una volta fatto, apri IMP, vai su File > Crea > Startrail. Imposta 'Light Frames' sulla cartella in cui hai messo tutte le foto che desideri utilizzare e fai clic su OK.

> "Potresti essere in grado di creare lo scatto perfetto"

03 Altre opzioni

Ci sono più opzioni con cui giocare, che puoi trovare sul sito Web del plug-in, che possono aiutarti a perfezionare la tua foto. Dagli un'occhiata se non sei soddisfatto del risultato, e potresti essere in grado di personalizzare lo scatto perfetto.

star_trails.py

> Linguaggio: **Python**

```
001. #!/usr/bin/python3
002. import time
003.
004. from picamera2 import Picamera2
005.
006. picam2 = Picamera2()
007. picam2.configure("still")
008. picam2.start()
009.
010. # Dai tempo ad Aec e Awb di sistemarsi, prima di
     disabilitarli. Tempo di esposizione di 30s
011. time.sleep(1)
012. picam2.set_controls({{"ExposureTime": 30000000,
     "AeEnable": False, "AwbEnable": False, "FrameRate": 1.0})
013. # E attendi che queste impostazioni abbiano effetto
014. time.sleep(1)
015.
016. start_time = time.time()
017. for i in range(1, 100): # 100 shots
018. r = picam2.capture_request()
019. r.save("main", f"image{i}.jpg")
020. r.release()
021. time.sleep(300) # 5 minuti di ritardo per il timelapse
022.
023. picam2.stop()
```

LOCATION, LOCATION, LOCATION

Ovviamente vorrai scattare di notte, ma da dove? Il posto migliore è da qualche parte lontano da una città o area urbana, dove l'inquinamento luminoso non pregiudichi la vista delle stelle. Il sito web **gostargazing.co.uk** è eccellente per consigli sui luoghi da visitare (in UK, NdZzed)

Scattare con un telescopio

Ottieni nitide foto di oggetti distanti utilizzando un telescopio e un software intelligente

Per scattare foto meravigliose dei corpi celesti, devi essere in grado di mantenerli in vista, il che significa muovere la fotocamera.

Fortunatamente, con il software giusto, hardware e un Raspberry Pi, puoi ottenere alcune foto straordinarie.

Hubble Pi

Lo studente Santiago Rodriguez ha utilizzato un semplice approccio per l'installazione di un telescopio per astrofotografia con Raspberry Pi – ha semplicemente utilizzato un adattatore per l'oculare del telescopio, per la HQ Camera.

Fatto ciò, ha creato un'interfaccia utente speciale chiamata AstroCam che gli permette di puntare il telescopio nel punto giusto e poi anche controllare alcune delle impostazioni della fotocamera, come quelle discusse in precedenza. È un risultato molto più economico.

"Le migliori fotocamere USB per l'astro-fotografia partono da circa 200€ e richiedono un computer connesso in ogni momento. Anche Hubble Pi può funzionare in modalità wireless o come standalone, se necessario", ci ha detto quando lo abbiamo intervistato a riguardo.

magpi.cc/hubblepi

PiFinder

Molti telescopi moderni possono connettersi al computer per aiuto nel rintracciare oggetti distanti. Richard (alias "brickbots") ha fatto un sacco di controllo manuale con la carta e poi un computer commerciale - ha deciso di costruirne uno basato su Raspberry Pi.

Utilizza una combinazione di dati di posizione GPS e un feed visivo da una telecamera HQ per capire esattamente dove è puntato il telescopio, comprende un'interfaccia personalizzata in modo de essere controllata in campo (letteralmente).

"La mia speranza è che altre persone trovino utile la combinazione di funzionalità, costruirà il proprio PiFinder e aiuterà l'intero progetto migliorare dando suggerimenti e potenzialmente contribuendo al software. È abbastanza semplice, costruito con parti standard e adatto ai principianti Della saldatura", dice Richard sul suo sito web.

magpi.cc/pifinder

Autoguida per astrofotografia

Dopo aver passato molte notti cercando di ottenere grandi scatti da sequenze di foto – e troppi non usciti molto bene – Joe Kutner ha deciso di programmare la propria autoguida che può mantenere il suo telescopio puntato sullo stesso oggetto celeste.

> Ogni passo del processo ha avuto le sue sfide

Utilizzando un touchscreen e Raspberry Pi, Joe ha creato un sacco di script automatizzati che possono essere attivati con la semplice pressione di un pulsante, e ora è certo che il suo telescopio punta sempre lo stesso oggetto nel cielo notturno.

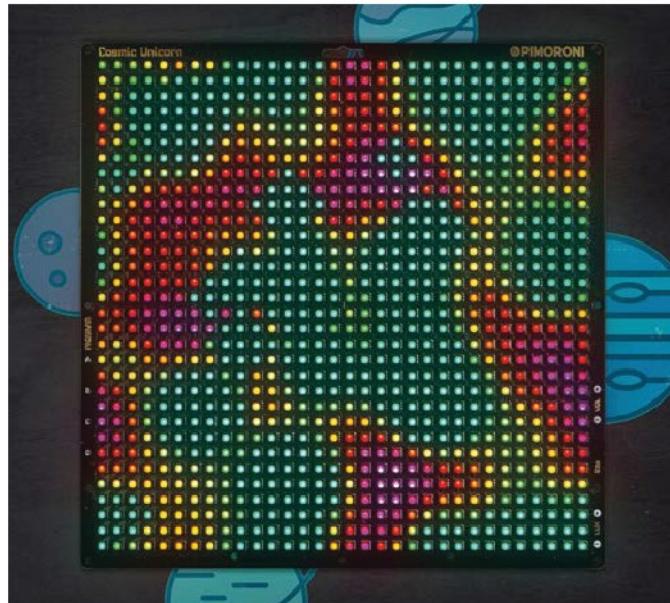
"Ogni fase del processo ha avuto le sue sfide", ci ha detto Joe qualche anno fa. "Installavo un software e poi scoprire che non era compatibile con una versione di un altro software di cui avevo bisogno. Quando finalmente ho ottenuto tutto funzionante, non parlava con il mio telescopio finché non ho installato un'altra versione del software. C'erano dozzine di questi piccoli ritagli di carta ma, alla fine, è valsa la pena lavorarci sopra.

magpi.cc/autoguider

Vuoi sapere qual'è il periodo migliore per vedere un oggetto specifico? Il calendario di Sea and Sky ha date per tutti i principali eventi del cielo notturno. Guarda: **magpi.cc/astrocal**.

10 Fantastici: Monitor e display

Guarda cosa state facendo con questi add-on incredibilmente diversi

Usaredi un Raspberry Pi senza alcun tipo di schermo è abbastanza comune. Tuttavia, se vuoi mostrare qualcosa visivamente da Raspberry Pi, o anche da Raspberry Pi Pico, hai solo l'imbarazzo della scelta, non solo per quanto riguarda le dimensioni ma anche per tipo. Ecco dieci dei nostri preferiti.

▲ HyperPixel 2.1 Rotondo

Cerchio HD

Un minuscolo schermo che si adatta perfettamente a un Raspberry Pi Zero. È probabilmente troppo grande per essere usato come orologio, ma nessuno ti fermerà se ci stai provando.

magpi.cc/hyperpixel2r | 51€ / 65$

▲ Cosmic Unicorn

LED a 1024 colori

La linea di matrici di LED Unicorn è un'ottima alternativa a un display tradizionale - e con così tanti LED, il Cosmic unicorn potrebbe anche essere un vero e proprio display.

magpi.cc/cosmicunicorn | 76€ / 97$

▲ EPD Pico Development Kit

Scheda e-ink

Questo add-on ti consente di collegare un display e-ink a Pico. È su un cavo, quindi Pico non deve esserlo montato su di esso

magpi.cc/epdk | 30€ / 30$

▼ IPS wide touchscreen

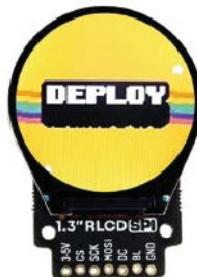
Ragazzaccio largo

Se hai bisogno di un display molto ampio o molto lungo, questo touchscreen da 7,9 pollici con un formato 16:5 è un ottimo inizio

magpi.cc/ipswide | 81€/ 103$

▲ HyperPixel 4.0

Touchscreen da 4 pollici

Quello che consideriamo il piccolo display standard per un Raspberry Pi, l'HyperPixel 4.0 è in circolazione da un po' ed è ancora il migliore della sua classe.

magpi.cc/hyperpixel | 51€ / 65$

▼ 1.3˝ OLED

Piccolo display

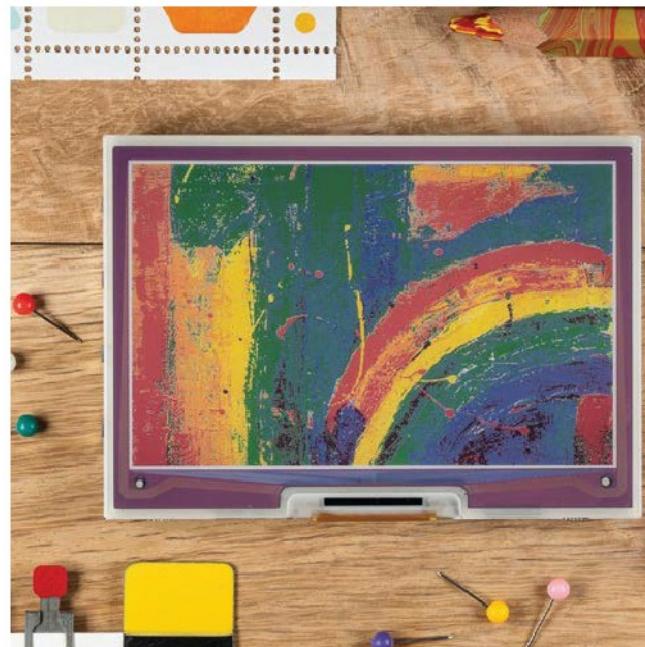
Questo minuscolo schermo si adatta perfettamente a un Pico, con 64×128 pixel con cui giocare. Ha anche un paio di pulsanti!

magpi.cc/oled13 | 10€ / 11$

▲ Scheda con LCD rotondo

Schermo a colori SPI

Questo piccolo schermo rotondo si collega a un Pimoroni Breakout Garden, inserendosi in una delle tant e porte presenti.

magpi.cc/roundlcd | 19€ / 30$

▼ GFX HAT

Fascino vecchia scuola

Abbiamo un debole per il GFX HAT, offrendo un display LCD retro illuminato con sei pulsanti capacitivi.

magpi.cc/gfxhat | 13€ / 21$

▲ Inky Impression

Add-on e-paper

Questo display a sette colori ha un bell'aspetto e consuma pochissima energia per funzonare. Ottimo per progetti più artistici.

magpi.cc/inkyimpression | Da 57€ / 60$

▲ Raspberry Pi Touch Display

Pronto per la scrivania

Il touchscreen ufficiale si inserisce nella porta DSI su Raspberry Pi full-size e l'insieme può essere reso abbastanza compatto, se necessario.

magpi.cc/touch | 73€ / 83$